FAST, ITERATIVE SYSTEM AND METHOD FOR EVALUATING A MODULO OPERATION WITHOUT USING DIVISION

BACKGROUND OF THE INVENTION

[0001] This application claims the benefit of provisional U.S. Application Serial No. 60/316,135, entitled "FAST, ITERATIVE SYSTEM AND METHOD FOR EVALUATING A MODULO OPERATION WITHOUT USING DIVISION," filed August 29, 2001, which is incorporated herein by reference in its entirety for all purposes.

Field of the Invention:

[0002] The present invention relates to electronic circuits and systems. More specifically, the present invention relates to hardware implementation of arithmetic operators for use in communications systems.

Description of the Related Art:

[0003] Interleaving of coded data for transmission (in combination with deinterleaving at the receiver) has been an effective method of transforming burst errors into statistically independent errors. Interleaving reorders the coded data sequence in an apparently random order, such that after the data is returned to its proper sequence by the deinterleaver, error bursts are spread out in time. Thus errors within one code word appear to be independent.

[0004] Previous transmission standards used a method of interleaving involving bit reversal of parts of the binary representation of bin numbers to randomize the data sequence. However, a new wireless standard, CDMA2000, requires a deinterleaver which uses a modulo operation. In particular, it requires the evaluation of M modulo J for $0 < M < 2^N$ and J = 3, 6, 12, 24, 48, and 96. The operation M modulo J returns the remainder of M divided by J. Currently, there is no hardware design which implements this modulo operation.

[0005] Hence, a need exists in the art for a fast system and method for evaluating M modulo J which can be easily implemented in hardware.

SUMMARY OF THE INVENTION

[0006] The need in the art is addressed by a technique which provides a fast, iterative method for evaluating M modulo J (M mod J) which can be easily implemented in hardware for use in such applications as deinterleavers in communications systems.

[0007] In an illustrative implementation, the invention includes the steps of: 1) decomposing M into two integers A and B = M – A; 2) evaluating C = A modulo J; 3) evaluating M′ = C + B; and 4) determining whether to output M′ as the final answer, or to feedback M′ to said first means to evaluate M′ modulo J.

[0008] The method may be easily implemented in hardware where for example in Step 1, the integer A is a power of 2 and, in Step 2, A modulo J is stored in a small look-up table for $A = 2^0, 2^1, 2^2 \ldots 2^N$.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] FIG. 1 is a block diagram of a typical wireless communications system.

[0010] FIG. 2 is a block diagram of a typical bit-reversal order deinterleaver.

[0011] FIG. 3 is a flow diagram of an iterative algorithm for evaluating M modulo J in accordance with the teachings of the present invention.

[0012] FIG. 4 is a block diagram of a hardware implementation for evaluating M modulo J in accordance with the teachings of the present invention.

DESCRIPTION OF THE INVENTION

[0013] Illustrative embodiments and exemplary applications will now be described with reference to the accompanying drawings to disclose the advantageous teachings of the present invention.

[0014] While the present invention is described herein with reference to illustrative embodiments for particular applications, it should be understood that the invention is not limited thereto. Those having ordinary skill in the art and access to the

teachings provided herein will recognize additional modifications, applications, and embodiments within the scope thereof and additional fields in which the present invention would be of significant utility.

[0015] FIG. 1 is a block diagram of a typical communications system 200 using a deinterleaver. The cdma2000 transmission standard calls for an interleaver and a deinterleaver which evaluate M modulo J for $0 < M < 2^N$ and J = 3, 6, 12, 24, 48, and 96. The operation M modulo J (M mod J) returns the remainder of M divided by J.

[0016] FIG. 2 is a block diagram of a typical bit-reversal order deinterleaver 94. The deinterleaver includes a demultiplexer 102, a multiplexer 106, and a circuit 104 for evaluating $A_i = 2^m (i \bmod J) + BRO_m(\lfloor i/J \rfloor)$, where $\lfloor x \rfloor$ indicates the largest integer less than or equal to x, $BRO_m(y)$ indicates the bit-reversed m-bit value of y (for example, $BRO_3(6) = 3$), and m and J are given in the following table for a deinterleaver of size N:

Interleaver Size	m	J
48	4	3
96	5	3
192	6	3
384	6	6
768	6	12
1,536	6	24
3,072	6	18
6,144	7	48
12,288	7	96
144	4	9
288	5	9
576	5	18
1,152	6	18
2,304	6	36
4,608	7	36

9,216	7	72
18,432	8	72
36,864	8	144
128	7	1

[0017] The symbols input to the deinterleaver 94 are written sequentially at addresses 0 to N-1 in the demultiplexer 102. At the output of the deinterleaver, the symbols are read out in permuted order from address A_i, for $i = 0$ to N-1. The circuit 104 evaluates A_i, and the multiplexer 106 combine the symbols sequentially from A_0 to A_{N-1}.

[0018] In computing A_i, the circuit 104 needs to evaluate M modulo J for $M = 2^m i$ for $i = 0$ to N-1, and m and J given by the above table. Ideally, the M modulo J operation should be implemented in hardware.

[0019] The present invention provides a fast, iterative method for evaluating M modulo J (M mod J) which can be easily implemented in hardware for use in applications such as the deinterleaver described above.

[0020] A recursive formula for computing M modulo J can be derived from the following algebraic manipulations:

[0021] Let M be an integer from 0 to 2^N. M can be expressed as a sum of two other integers:

[0022] $M = A + B.$ (1)

[0023] For any integer J, there exists unique integers q_a, q_b, r_a, and r_b such that:

[0024] $A = q_a J + r_a$,

[0025] and

[0026] $B = q_b J + r_b$, (2)

[0027] where $J > r_a, r_b > 0$.

[0028] Therefore,

[0029] $A + B = (q_a + q_b)J + (r_a + r_b)$ (3)

[0030] It follows that:

[0031] M modulo J = (A + B) modulo J

[0032] = $[(q_a + q_b)J + (r_a + r_b)]$ modulo J

[0033] = $(q_a + q_b)J$ modulo J + $(r_a + r_b)$ modulo J

[0034] = $(r_a + r_b)$ modulo J, (4)

[0035] since $(q_a + q_b)J$ is an integer multiple of J, and therefore has a remainder of 0 when divided by J. Thus $(q_a + q_b)J$ modulo J is equal to 0. By similar reasoning, adding a term which is an integer multiple of J will not affect the modulo J operation:

[0036] $(r_a + r_b)$ modulo J = $(r_a + q_bJ + r_b)$ modulo J

[0037] = [(A modulo J) + B] modulo J, (5)

[0038] since A modulo J = $(q_aJ + r_a)$ modulo J = r_a.

[0039] Therefore:

[0040] M modulo J = M′ modulo J, (6)

[0041] where M′ = A modulo J + B. This leads to an iterative algorithm for evaluating M modulo J.

[0042] FIG. 3 is a flow diagram of an iterative algorithm for evaluating M modulo J in accordance with the teachings of the present invention. This method includes the following steps:

[0043] decomposing M into two integers A and B = M – A;

[0044] evaluating C = A modulo J;

[0045] evaluating M′ = C + B; and,

[0046] determining whether to output M′ as the final answer, or to repeat with M = M′ to evaluate M′ modulo J.

[0047] This method can be readily implemented in hardware if in Step 1, the integer A is a power of 2, and in Step 2, A modulo J is stored in a look-up table for A = $2^0, 2^1, 2^2 \ldots 2^N$. Let M be an integer in binary representation, that is, $M = \sum \alpha_i 2^i$ for i = 0 to N, and $\alpha_i = 0$ or 1. In step 1, A is chosen to be $\alpha_i 2^i$. Since α_i is either 0 or 1, A is either 0 or 2^i. Thus in Step 2, A modulo J is 0 for $\alpha_i = 0$, or 2^i modulo J for $\alpha_i = 1$. Then, 2^i modulo J can be evaluated through a small look-up table storing 2^i modulo J for i = 0 to N, and the values of J required (for this particular application, J = 3, 6, 12, 24, 48, or

96). The algorithm is repeated recursively, starting with i = N, and reducing i by 1 with each iteration, until a final answer is reached when $M' < J$.

[0048] In one case, the algorithm does not converge. An additional step between Step 3 and Step 4 is required to insure convergence to the correct answer:

[0049] Step 3.5: if the bitwise AND between M′ and J equals J, then let $M = M' - J$ and return to Step 1, otherwise output M′ as the final answer.

[0050] The following is a numerical example to further illustrate this method:

[0051] EXAMPLE: Find M modulo J for M = 27, J =6.

[0052] $M = 11011_{bin} = 2^4 + 2^3 + 2^1 + 2^0$

[0053] Let $A = \alpha_4 2^4 = 10000_{bin} = 16$, and $B = M - A = 1011_{bin} = 11$

[0054] From a look-up table, find C = A modulo J = 16 modulo 6 = 4

[0055] Form $M' = C+B = 4+11 = 15 = 1111_{bin}$

[0056] Step 3.5: Check if $(M' \&\& J = J)$: 1111_{bin} && 110_{bin} = 110_{bin} = J, therefore let $M' = M' - J = 15-6=9$

[0057] Step 4: Check if $(M' < J)$: $9 > 6$, therefore let $M = M' = 9$

[0058] and repeat

[0059] Step 1: $M = 9 = 1001_{bin}$

[0060] Let $A = 1000_{bin} = 8$, and B = M-A = 1

[0061] Step 2: From a look-up table, find C = A modulo J = 8 modulo 6 = 2

[0062] Step 3: Form $M' = C+B = 2+1 = 3 = 11_{bin}$

[0063] Step 3.5: Check if $(M' \&\& J = J)$: 11_{bin} && 110_{bin} = $10_{bin} \neq J$, therefore continue to Step 4

[0064] Step 4: Check if $(M' < J)$: $3 < 6$, therefore stop. The final answer is 3.

[0065] Therefore, 27 modulo 6 = 3.

[0066] FIG. 4 is a block diagram of an illustrative hardware implementation for evaluating M modulo J in accordance with the teachings of the present invention. The architecture includes a first circuit 10 for decomposing M into two integers A and B = M – A (STEP 1); a second circuit 20 for evaluating A modulo J (STEP 2); a third circuit 30 for evaluating M′ = (A modulo J) + B (STEP 3); a fourth circuit 40 for determining whether to output M′ as the final answer, or to feedback M′ to the first circuit 10 to

evaluate M′ modulo J (STEP 4); and, a fifth circuit 50 for ensuring convergence (STEP 3.5).

[0067] The inputs to this circuit are two integers M and J. Initial conditions are set such that $i = N$, and $B_N = M - \alpha_N 2^N$.

[0068] The first circuit 10 includes a multiplexer M1 which passes $B_N = (M - \alpha_N 2^N)$ on the first iteration, and passes $B_i = (M' - \alpha_i 2^i)$ on all subsequent iterations, where i is an iteration counter starting with N and counting down. The output of the multiplexer M1 (equivalent to B in the derivations) is passed to the third circuit 30.

[0069] The second circuit 20 includes a look-up table 22 which stores 2^i modulo J for i = 0 to N. The second circuit 20 further includes a multiplexer M2 which passes 0 if ($\alpha_i = 0$), and passes C_i if ($\alpha_i = 1$). The output of M2 is therefore equivalent to A modulo J, where $A = \alpha_i 2^i$. This output is passed to the third circuit 30.

[0070] The third circuit 30 includes an adder A1 which adds the outputs of the first and second circuits and passes the result M′ = (A modulo J) + B to the fifth circuit 50.

[0071] The fifth circuit 50 includes a multiplexer M3 which passes J if the bitwise AND of M′ and J equals J, otherwise it passes 0. The output of M3 is subtracted from M′ by an adder A2, and the result is passed to the fourth circuit 40.

[0072] The fourth circuit 40 includes a multiplexer M4 which passes M′ as the final output if ($M' < J$); otherwise i is set to i-1, and M′ is fed back to the first circuit 10. The feedback loop is repeated until the condition $M' < J$ is met. Then M′ is output as the final solution to M modulo J.

[0073] Hence, the new hardware implementation of FIG. 3 evaluates the operation M modulo J.

[0074] Thus, the present invention has been described herein with reference to a particular embodiment for a particular application. Those having ordinary skill in the art and access to the present teachings will recognize additional modifications, applications and embodiments within the scope thereof. For example, those skilled in the art will appreciate that for the algorithm can be used in applications other than a deinterleaver in a communications system. Further, the invention can be used in any digital signal processing (DSP) application requiring the operation M modulo J.

[0075] It is therefore intended by the appended claims to cover any and all such applications, modifications and embodiments within the scope of the present invention.

[0076] Accordingly,